OCR
Oxford Cambridge and RSA

An OCR endorsed teaching and learning tool

OCR A Level Computer Science

H446 – Paper 1

2

Queues

Unit 7
Data structures

PG ONLINE

Objectives

- Understand the concept of an abstract data type
- Be familiar with the concept and uses of a queue
- Describe the creation and maintenance of data within a queue (linear, circular, priority)
- Describe and apply the following to a linear, circular and priority queue
  - add an item
  - remove an item
  - test for an empty queue
  - test for a full queue

Introduction to data structures

- Sort these terms into the table based on your current understanding:
  - array, integer, real, list, stack, Boolean, char, queue, string

Category	Word
Elementary data type	
Composite data type	
Abstract data type	

Introduction to data structures

Category	Word
Elementary data type	integer, real, Boolean, char
Composite data type	string, array *[list may also be in this section, depending on programming language]*
Abstract data type	list, stack, queue

Abstraction

- An abstract data type (ADT) is a logical description of how we view the data and possible operations
  - A queue of print jobs (add to the rear, remove from front)
  - A stack of books (add to top, remove from top)
  - A list of tasks to do (add to the end, remove most important)
- We are concerned only with what the data is representing and not how it is constructed

Examples of queues

- What are some examples of queues in real life and information processing systems?
- What operations can be carried out on a queue?

Modelling a queue

- You might have thought of a queue at the cinema or supermarket checkout, or a queue of jobs waiting to be processed or printed
- There are four distinct operations:
  - Add item to the rear of the queue
  - Remove item from the front of the queue
  - Check if the queue is empty
  - Check if the queue is full

Add and remove

- Using **only** the bus cards and the queue card:
  - Add bus **118A Sheffield** to the queue
  - Add another bus **92B York** to the queue
  - Add bus **142A Leeds** to the queue
  - Remove a bus from the queue
  - Remove another bus from the queue

Add and remove

- After three buses join the queue, it looks like this:

[0]	[1]	[2]	[3]	[4]
118A Sheffield	92B York	142A Leeds		

- After two buses leave the queue, it looks like this:

[0]	[1]	[2]	[3]	[4]
		142A Leeds		

- Is this a good way of implementing a queue?

Front and rear

- It’s very wasteful of CPU cycles to refill memory locations with blanks - pointers can be used instead
  - What should the front pointer hold after three buses have joined, and two buses have left the queue?
  - What should the rear pointer hold?

[0]	[1]	[2]	[3]	[4]
118A Sheffield	92B York	142A Leeds		

front = ?

rear = ?

Front and rear

- There are different ways of implementing a queue
  - The choice only affects the implementation of the operations
  - Recall that the implementation is abstracted away

[0]	[1]	[2]	[3]	[4]
118A Sheffield	92B York	142A Leeds		

front = 2 rear = 2

- front points to the next item to remove and rear points to the last item added

Empty and full queues

- We can't add to a full queue or remove an item from an empty queue
- Therefore, when the queue is initialised, we need to specify the maximum number that it can hold, e.g. maxSize
- We may also need a variable size to hold the number of items currently in the queue
  - How will a full queue be detected?

Queue functions or methods

- enQueue(item) – add an item to the rear
- deQueue – remove and return an item from the front
- isEmpty – indicates if the queue is empty
- isFull – indicates if the queue is full

Worksheet 2

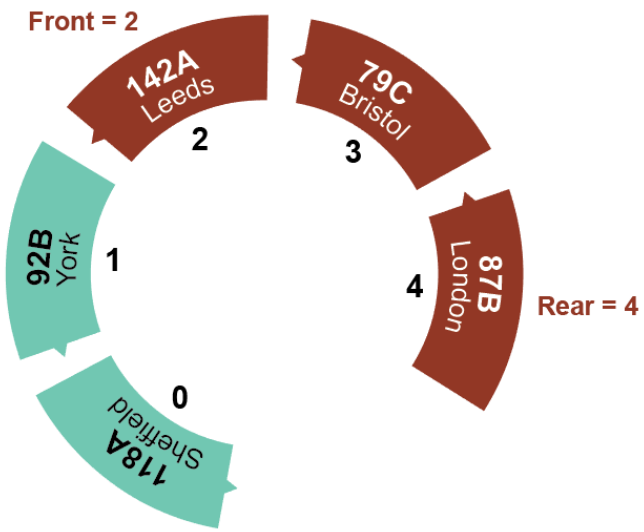
- Complete **Task 1**
  - Mochi (shown here) are traditional Japanese rice cakes!

Problems with implementation

- Problems arise with the implementation of a queue as a fixed-size array
- How many items can be added? Or removed?
- What happens when the queue is full, but there are some free spaces at the front?
  - How could these limitations could be overcome when using an array?

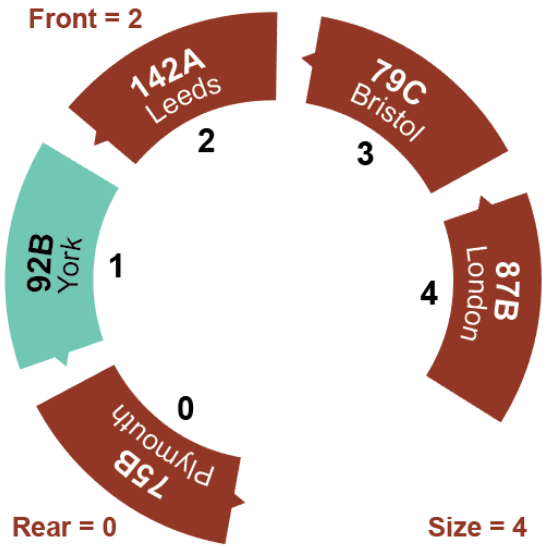
Circular queue

- A circular queue algorithm overcomes the problem of reusing the spaces that have been freed by dequeueing from the front of the array



- Bus 75 from Plymouth pulls in. Where will it go? Adjust the pointers

Adding to a circular queue



- Pointers go: 0 → 1 → 2 → 3 → 4 → 0 → 1 …
- What function can you use to implement this?
  - How will you test for a full queue?

MOD function

- Complete to show the operation of the MOD function

Current Index	(Current + 1)	(Current + 1) MOD 5	New Current Index
0			
1			
2			
3			
4			

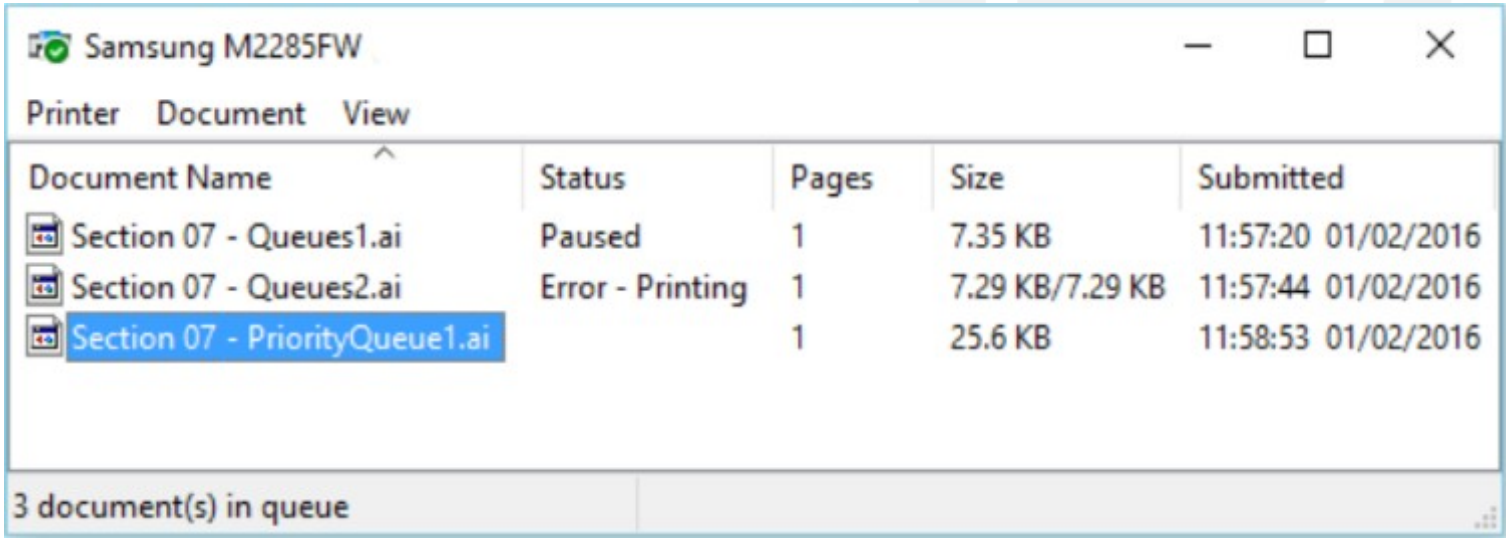
MOD function

- Complete to show the operation of the MOD function

Current Index	(Current + 1)	(Current + 1) MOD 5	New Current Index
0	1	1	1
1	2	2	2
2	3	3	3
3	4	4	4
4	5	0	0

Worksheet 2

- Complete **Task 2** on the worksheet



Pseudocode

- enqueue(item) – add an item to the rear
- dequeue – remove and return an item from the front
- isEmpty – indicates if the queue is empty
- isFull – indicates if the queue is full

Functions isEmpty, isFull

- How do you know if the queue is empty?
- How do you know if the queue is full?
- Assume a queue has been defined as:

```
q = array of maxSize elements
front = 0
rear = -1
size = 0
```

- Write pseudocode for isEmpty and isFull

isEmpty, and isFull

```
function isEmpty
  if size == 0
then
    return True
  else
    return False
  endif
endfunction
```

```
function isFull
   if size == maxSize
then
      return True
   else
      return False
   endif
endfunction
```

- Can you write these subroutines in a shorter way?

Adding and deleting elements

```
procedure enqueue(newItem)
  if size == maxSize  then
    print ("Queue full")
  else
    rear = (rear + 1) MOD maxItems
    q[rear] = newItem
    size = size + 1
  endif
endprocedure
```

- Write a subroutine to dequeue an item

Priority queue

- In a priority queue, some items are allowed to ‘jump’ the queue
- This type of queue is used when the items arriving have some type of priority associated with them
  - What types of priority queues exist in real-life?

Priority queue

- Each item has a priority associated with it
- In the example of the buses, assume that:
  - Buses with numbers ending with **A** are high priority
  - Buses with numbers ending with **B** are medium priority
  - Buses with numbers ending with **C** are low priority
- Ignoring the complications of a circular queue, insert the following buses into a priority queue:

  92B, 64C,142A, 25C, 87B

  - What order are your buses in now?

Priority queue

- You have added the buses **92B**, **64C**, **87B**, **25C** and **142A** to the queue

142A	92B	87B	64C	25C

- The buses in the queue are in order of priority
- What algorithm are you using to insert buses into the queue?
- **142A** leaves and **75B** joins the queue.
  - What does the queue look like now?

Dynamic vs static

- Static data structures are fixed in size
  - Cannot grow, shrink, or be freed during execution
  - An array is a static data structure
- Dynamic data structures can grow and shrink
  - Allocates and deallocates memory from the **heap** (an area of memory especially used for this purpose)
  - Excessive allocation of memory, without deallocation, may cause overflow (exceeding maximum memory limit)
  - Python – list; Java - ArrayList

Worksheet

- Complete the ‘Accident and Emergency’ **Task 3** on the worksheet

Plenary

- Fill in the following table showing a comparison between a simple linear array queue, a circular queue, and a circular priority queue

FIFO	Queue (array)	Circular Queue	Priority Queue
Advantages			
Disadvantages			
Usage			

Answers

- Fill in the following table showing a comparison between a simple linear array queue, a circular queue, and a circular priority queue

FIFO	Queue (array)	Circular Queue	Priority Queue
Advantages	Simple to program Predictable memory usage	Can reuse free spaces	Gives preference to more important items
Disadvantages	Fixed length Single pass Can't reuse spaces	Slightly more complex to program	Additional processing required to keep order
Usage	Pre-board on a roller-coaster	Printer spooler	Accident & emergency

Extension

- Changing the rules to serve the last person to arrive first, reduces the average wait time
- Why does the queue **you're not in** always move faster?
  - http://sciencenordic.com/queues-move-faster-if-last-person-served-first
  - http://www.bbc.co.uk/news/magazine-34153628

Copyright

Licence agreement

PG ONLINE